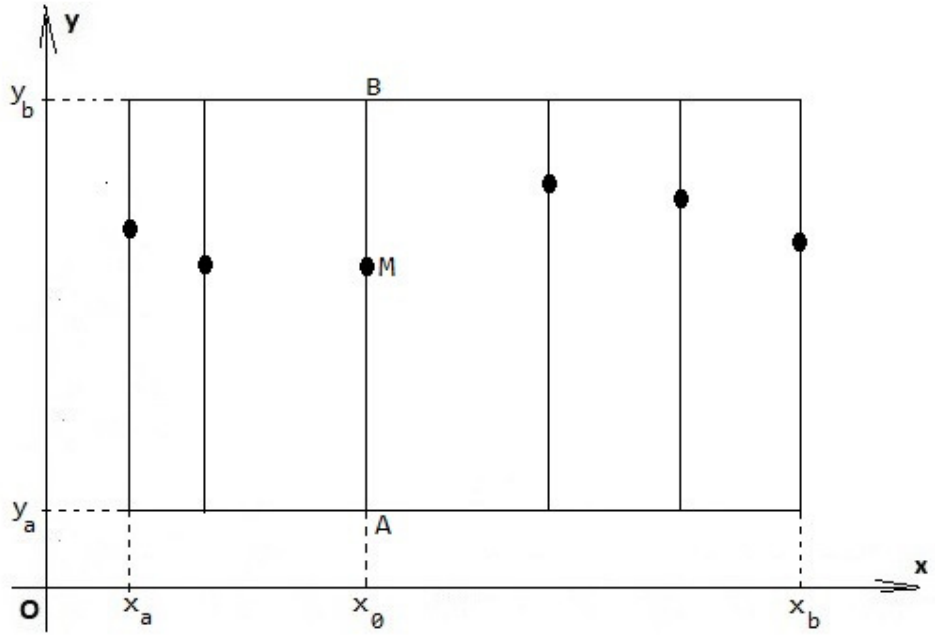
Минимизация функции двух переменных.

Кажется, это самый непростой вопрос этого занятия, хотя он довольно простой и ясный, если въехать. Но надо это всё компактно уложить в голове, и тогда всё становится действительно простым до очевидности. Для этого надо приложить некоторые усилия, но это обычное дело - укладывание в голове требует усилий. Да и мы с таким уже сталкивались не раз.

Метод, повторю, довольно естественный – сведём задачу поиска минимума функции двух переменных к задаче поиска минимума функции одной переменной. Разумеется, за такие свдения к более простой задаче надо чем-то платить. И плата здесь будет такая, что мы будем искать минимум функции одной переменной многократно, но всё-таки умеренно много, да мы и не особо будем сегодня заботиться об эффективности, в разумных пределах, конечно.

Собственно, главные слова уже сказаны – сводим двумерную задачу к одномерной. Причём довольно простым способом.



Итак, ищем минимум некоторой функции f(x, y). Взглянем на рисунок. Там выделен прямоугольник $x_a \le x \le x_b$, $y_a \le y \le y_b$. Это та область, в которой мы ищем минимум. На рисунке изображена только плоскость xOy, подобно тому, как мы изображали только ось Ox, когда строили методы одномерной минимизации (одномерной – имеется в виду минимизация функции одной переменной). Нас интересуют только передвижения по аргументам функции, а значения функции мы, да, будем сравнивать, но это и всё, они нам на рисунке не очень-то и нужны.

Давайте зафиксируем какой-нибудь x из интервала $[x_a, x_b]$, обозначим его x_0. Пробежимся по отрезку от точки $A(x_0, y_a)$ до точки $B(x_0, y_b)$, вычисляя в каждой точке f(x, y). Мысленно, конечно, ведь точек бесконечно много. Что мы получаем в итоге? Для всех точек $x = x_0$, т.е. x не изменяется, изменяется при этом только y. Иначе говоря, мы вычисляли значения некоторой функции от y, назовём её q: $q_{x0}(y) = f(x_0, y)$. $q_{x0}(y)$ есть функция одной переменной y (x_0 в ней играет роль параметра и, повторю, не изменяется). Мы можем найти точку минимума функции $q_{x0}(y)$ – на рисунке она выделена и обозначена M – ведь мы умеем находить точку минимума одномерной функции. Точка M зависит только от x_0 (ну, и от функции f, конечно), т.е. величина $f(M) = f(x_M, y_M) = f(x_0, y_M)$ полностью определяется величиной x_0. f(M) – есть наименьшее значение функции f при $x = x_0$, и полностью

определяется величиной x_0, т.е. f(M) = h(x_0), где h – некоторая одномерная функция, функция переменной x: h(x) – это минимальное значение функции f(x, y) среди всех y от y_a до y_b. Одномерная функция! Ну, так и вычислим её минимум. Понятно, что это и есть минимум функции f на всём прямоугольнике. Всё!
Давайте посмотрим, что же у нас получилось. Мы свели вычисление минимума двумерной функции f к вычислению минимума одномерной функции h. При этом вычисление одного значения функции h требует решения задачи одномерной минимизации, т.е. задачу одномерной минимизации нам надо решать многократно. Вот вам и ответ на вопрос, зачем нам было выжимать эффективность из методов одномерной минимизации, зачем мы вращались с методами Фибоначчи и/или "золотого сечения".

Для большей ясности я приведу программу на Go, реализующую этот алгоритм:

```
package main

import "fmt"

type (
     Function1D func(float64) float64
     Function2D func(float64, float64) float64
)

func minimum2D(left, right, bottom, top float64, F2 Function2D)
(float64, float64) {
     h:= (func (x float64) float64 {
          q:= (func (y float64) float64  {
               return F2(x, y)
          } )
          ymin:= minimum1D(bottom, top, q)
          return F2(x, ymin)
     }  )

     xmin:= minimum1D(left, right, h)
     ymin:= minimum1D(bottom, top, ( func (y float64) float64 {
                                        return f(xmin, y)
                                   }  )  )
     return xmin, ymin
}

func minimum1D(start, finish float64, F1 Function1D) float64 {
     // Any procedure of the 1D minimization
     var pmin float64    //minimum point
     // ...
     return pmin
}

func f(x, y float64) float64 {
     var res float64
     // ...
     return res
}

func main() {
     var left, right, bottom, top float64
     // left, right, bottom, top = ...
     fmt.Println(minimum2D(left, right, bottom, top,  f))
}
```

Конечно, это не совсем программа, это заготовка, в которую надо включить какую-нибудь процедуру одномерной минимизации, а также определить функцию f и прямоугольник, на котором мы отделили её минимум: left и right – это x_a и x_b на рисунке 4, bottom и top – это y_a и y_b.

Остаётся вопрос о том, какими свойствами должен обладать прямоугольник отделения минимума функции f – тот самый прямоугольник, который на рисунке 4. Не будем это подробно обсуждать, скажем только что в этом прямоугольнике функция f должна быть дважды унимодальной (это означает, что если мы фиксируем x – так, как мы это сделали – то получающаяся одномерная функция q(y) должна быть унимодальной для каждого x; и то же самое должно происходить, если мы фиксируем y и получаем одномерную функцию, зависящую от x). А ещё скажем, что это условие не страшное, и для функций g_1, g_2 и g_∞ (напомню, если вы забыли – давно это было – что это те самые функции, которые мы используем для оценки отличия аппроксимирующей функции от измеренных данных) выполняется практически всегда, если только точка минимума попадает в прямоугольник. А точка минимума – это и есть искомые параметры аппроксимирующей функции, их мы, как правило, приблизительно, пусть даже очень грубо, можем оценить из физического (или какого-нибудь ещё) смысла задачи.

Задачи и упражнения.

I. Однопараметрическая аппроксимация.

"Резистор"

Даны результаты измерений силы тока I в амперах для восьми различных значений напряжения U в вольтах:

Напряжение, U (В)	10,0	20,0	30,0	40,0	50,0	60,0	70,0	80,0
Сила тока, I (А)	0,34	0,68	1,01	1,33	1,66	2,02	2,28	2,62

Напряжение и сила тока подчиняются закону Ома: $I = a \cdot U$, где $a = 1/R$, R – сопротивление резистора, a – проводимость резистора.

Определить проводимость резистора для трёх целевых функций

$$g_1(a) = |a \cdot U_1 - I_1| + |a \cdot U_2 - I_2| + ... + |a \cdot U_N - I_N|$$

$$g_2(a) = (a \cdot U_1 - I_1)^2 + (a \cdot U_2 - I_2)^2 + ... + (a \cdot U_N - I_N)^2$$

$$g_\infty(a) = \max\left(|a \cdot U_1 - I_1|, |a \cdot U_2 - I_2|, ..., |a \cdot U_N - I_N|\right)$$

Входные данные находятся в файле `01.resistor.dat`.

"Ускорение свободного падения". Для определения ускорения свободного падения g ($м/с^2$) измерены расстояния, которые пролетел свободно падающий груз:

Время, t (сек.)	0.2	0.4	0.6	0.8	1.0
Расстояние, h (м)	0.1960	0.7850	1.7665	3.1405	4.9075

Найти g для трёх целевых функций g_1, g_2 и g_∞, исходя из соотношения

$$h = gt^2/2.$$

Извините, но ускорение свободного падения традиционно обозначается той же буквой g, что я выбрал для целевых функций, но, надеюсь, это не приведёт к путанице.

Входные данные находятся в файле `01.gravitation.dat`.

II. Двупараметрическая аппроксимация.

"Камень". Вертикально вверх вылетает камень со скоростью v_0, сопротивление воздуха считаем пренебрежимо малым. Высота камня h спустя время t от начала полёта определяется соотношением

$$h = v_0 t - gt^2/2,$$

где g – ускорение свободного падения.

В таблице приведены результаты измерений местоположения (высоты) камня в зависимости от времени :

Время t, сек	Высота h, м	Время t, сек	Высота h, м	Время t, сек	Высота h, м
0.0	0.00	1.6	20.53	3.2	16.00
0.2	3.99	1.8	21.35	3.4	13.67
0.4	7.51	2.0	21.79	3.6	10.98
0.6	10.65	2.2	21.84	3.8	7.82
0.8	13.41	2.4	21.40	4.0	4.31
1.0	15.81	2.6	20.66	4.2	0.42
1.2	17.73	2.8	19.49		
1.4	19.33	3.0	17.96		

Найти параметры v_0 и g, аппроксимируя данную зависимость.

Входные данные находятся в файле `02.stone.dat`.

"Река". В таблице приведены характеристики основных притоков некоторой реки:

Длина L, км	Водосбор s, км^2	Длина L, км	Водосбор s, км^2
498	25100	202	4440
100	1680	937	98200
93	1220	99	1250
226	4000	121	1800
170	1940	76	520
140	2060	115	1345
67	791	105	1220
98	1140	325	6990

Найти формулу (параметры a и b) вида $s(L) = a \cdot L^b$, аппроксимирующую данную зависимость.

Входные данные находятся в файле `02.river.dat`.